國立交通大學

網路工程研究所

碩 士 論 文

網路流量多樣性與程式涵蓋率之實驗分析

On Traffic Diversity and Code Coverage: An Experimental Analysis

研 究 生：邵俊賓

指導教授：林盈達　教授

中 華 民 國 一百零一 年 一 月

網路流量多樣性與程式涵蓋率之實驗分析

學生：邵俊賓　　　　　　　　　　　　　　　　指導教授：林盈達

國立交通大學資訊學院網路工程研究所

摘要

網路流量可以用於測試網路產品。網路流量可以在很多方面有區別，例如流量裡的使用者行為或是流量蒐集的網路區域大小。假設來說，用具有較大多樣性的流量來測試網路產品可以達到較高的測試涵蓋率。但是如何描述流量的多樣性仍是疑問。在本論文中，定義了流量多樣性以及提出計算多樣性與分析測試涵蓋率的方法。以具有不同數量封包及從不等大小的網段蒐集到的流量來對目標原始碼做測試，結果發現測試涵蓋度會隨著封包數量增加及網域變大而增加。以在 Snort 上面的實驗結果來說，流量來源一可以達到 32.2%的測試涵蓋率，而流量來源二可以達到 31.8%。在 Linux kernel 上面，流量來源一可以達到 8.16%的測試涵蓋率，而流量來源二可以達到 9.47%。隨著封包數量增加及網域變大，流量的豐富度也是隨之增加，但流量的平均度卻可能隨著來自特定用戶或是特定網路協定的流量暴增而減低，因而導致定義的流量多樣性不一定都是呈現增加的狀態。我們可以藉由定義的豐富度及平均度來對取得的網路流量有更多解讀。

關鍵字： 流量多樣性，程式涵蓋率，網路測試

On Traffic Diversity and Code Coverage: An Experimental Analysis

Student: Chun-Pin Shao **Advisor: Dr. Ying-Dar Lin**

Institute of Network Engineering

National Chiao Tung University

Abstract

Network traffic traces can be used to test network devices. Network traffic traces may differ in many aspects, such as the user's behaviors and the size of network. It is assumed that using more diverse traffic to test network devices could have larger code coverage. However, how to describe the diversity of traffic traces is still an issue. In this thesis, traffic diversity index is defined and a methodology for calculating traffic diversity and analyzing code coverage is proposed. Packet traces with different number of packets and collected from different size of network segments are used to test target programs. Experiment results show that code coverage increases with more number of packets or larger size of network segments. In experiments on Snort, packet source 1 can achieve 32.2% code coverage while packet source 2 achieves 31.8%, and on Linux kernel, packet source 1 can achieve 8.16% while packet source 2 achieves 9.47%. Richness which is the number of types in a packet trace would not decrease with a larger number of packets or larger size of network segments in packet traces. However, evenness which is the relative proportion of a type of packets among the packets in a packet trace may not increase correspondently since there would be traffic bursts from the same user or the same network protocol, thus the decrease of traffic diversity. In the view of richness and evenness, we can learn more information about packet traces.

Keywords: traffic diversity, code coverage, network test

Content

List of Figures

List of Tables

Chapter 1 Introduction

Network traffic traces can be used to test network devices. For example, NCTU Betasite [1] collects traffic from dormitories to test network devices. Network traffic traces differ in many aspects. The complexity of network traffic generated by various traffic generators is different from traffic captured from the real world. The user's behaviors on different traffic traces from different network segments or captured during different time periods are also different.

A major issue of how to describe the differences between network traffic traces is developed in this thesis. An idea referred to biological diversity is introduced. Biological diversity [2] is a degree of variation of life species within a given ecosystem for measuring the health of a given ecosystem. Greater diversity implies better health. Considering packets that belong to a defined type as individuals of a life species, the concept of biological diversity can be used to describe how diverse a network traffic trace is and traffic diversity is coined in this thesis. Furthermore, a traffic diversity index is needed to quantify the traffic diversity to alleviate the comparison of traffic traces. There are several well-known diversity indices such as Simpson's index [3], Shannon's index [4][5] and Renyi's index [6][7]. Those indices are compared and one of them is chosen for calculating traffic diversity index.

Network traffic traces are used for testing network devices. A major concern of using network traffic traces to test network devices is how much code coverage the network traffic traces can achieve. Code coverage [8] is a measure used in software testing. It is a degree to which the source code of program has been tested. It is assumed that using more diverse network traffic to test network devices could have larger test coverage and hence may trigger more defects.

In this thesis, network traffic traces will be compared underlying traffic diversity and verified if more diverse traffic can achieve larger code coverage. The models based on traffic diversity index will be defined by TCP/IP header fields (fixed-length header fields) and application header fields (variable-length header fields). The diversity index for each header field of a traffic trace will be calculated according to the defined models of traffic diversity. All diversity indices of a traffic trace will be formed by a diversity vector and then merged as a combined diversity index. The traffic diversity of traffic traces will be compared according to every single traffic diversity index and the combined traffic diversity index. Next, code coverage will be applied where source code needs to be instrumented with code coverage analysis tool. Traffic traces that differ in number of packets and network of segments will be used to measure the code coverage where the code coverage analysis tool is applied. Finally, the correlation between traffic diversity indices and test coverage will be verified.

The organization of this thesis is as follows. In chapter 2, related works including introductions to diversity index, well-known diversity indices and code coverage will be presented. In chapter 3, definitions and problem statement will be presented. System architecture and proposed methodology will be presented in chapter 4. Experiment results and observations will be presented in chapter 5. Finally, conclusion of this thesis will be presented in chapter 6.

Chapter 2 Related Works

In this chapter, we shall introduce basic terms of diversity index and three well-known diversity indices. We shall give an example to illustrate them. Code coverage will be introduced as well. Criteria of code coverage will be illustrated by an example.

2.1 Diversity Index

Overview

A diversity index is a statistic to measure the local members of a set consisting of various types of objects. It was first introduced in ecology [1] to measure the biodiversity in an ecosystem. It can also be applied in other areas, such as in economics to measure the distribution over sectors of economic activity in a region, and in information science to describe the complexity of a set of information. We shall use the terms of ecology to explain diversity index in the rest of this chapter.

There are two basic factors of measuring diversity index: species richness and species evenness [2]. Species richness is simply the number of species present in a system and makes no use of relative abundances. The more species present in a habitat, the richer the habitat. Species evenness is the relative abundance or proportion of individuals among the species.

To give an example, we sample two different fields for wildflowers, as shown in Table 1. Sample 1 consists of 300 daisies, 335 dandelions and 365 buttercups, while sample 2, 20 daisies, 49 dandelions and 931 buttercups. Both samples have the same richness, 3 species, and the same total number of individuals. However, the first sample has more evenness than the second. This is because the total number of individuals in the first sample is quite evenly distributed between the three species than the second.

Table 1: Examples of richness and evenness

	Numbers of individuals	
Flower species	Sample 1	Sample 2
Daisy	300	20
Dandelion	335	49
Buttercup	365	931
Total	1000	1000

Simpson's index

Simpson's index [3], in terms of ecology, takes into account the species richness as well as the species evenness. The Simpson's index *D* represents the probability that two randomly selected individuals in the habitat will belong to the same species.

The formula for calculating *D* is

$$D = \frac{\sum_{i=1}^{S} n_i(n_i-1)}{N(N-1)},$$

where N represents the total number of individuals of all species, n_i the number of individuals in species *i*, *S* the number of species. However, we often use $\widetilde{D}$=$1-D$ to present diversity intuitively. Index 0, represents no diversity of species, while index 1, infinite. The bigger $\widetilde{D}$ represents more diversity.

To give an example, consider the example in Table 1. We calculate diversity index for sample 1 and sample 2 as

$$\widetilde{D}(Sample\ 1) = 1 - \frac{300(300-1)+335(335-1)+365(365-1)}{1000(1000-1)} = 0.666,$$

$$\widetilde{D}(Sample\ 2) = 1 - \frac{20(20-1)+49(49-1)+931(931-1)}{1000(1000-1)} = 0.131.$$

Therefore, sample 1 is more diverse than sample 2 in the view of Simpson's index.

Shannon's index

The Shannon index *H* [4][5] takes into account the species richness and the species evenness as well. It is the information entropy of the distribution, treating species as symbols as their relative population sizes as the probability.

The formula for calculating *H* is

$$H = -\sum_{i=1}^{S} p_i \ln p_i,$$

where p_i represents the relative abundance of each species, calculating as $\frac{n_i}{N}$. n_i is the number of individuals in species I, N the total number of all individuals, S the number of species.

To give an example, consider the example in Table 1. We calculate diversity index for sample 1 and sample 2 as

$$H(Sample\ 1) = -(\frac{300}{1000} ln \frac{300}{1000} + \frac{335}{1000} ln \frac{335}{1000} + \frac{365}{1000} ln \frac{365}{1000})\text{=1.212,}$$

$$H(Sample\ 2) = -(\frac{20}{1000} ln \frac{20}{1000} + \frac{49}{1000} ln \frac{49}{1000} + \frac{931}{1000} ln \frac{931}{1000})\text{=0.293.}$$

Therefore, sample 1 is more diverse than sample 2 in the view of Shannon's index.

Renyi's index

Renyi's index [6][7] is a generalization of Shannon's index. The Renyi's index of order α is defined as

$$H_\alpha = \frac{1}{1-\alpha} ln\left(\sum_{i=1}^{S} p_i^\alpha\right),$$

where $\alpha \geq 0, \alpha \neq 1$, p_i represents the relative abundance of each species, calculating as $\frac{n_i}{N}$, n_i the number of individuals in species I, N the total number of all individuals and S the number of species. Lower value of α, approaching zero, give a index which increasingly weights all possible events more equally, regardless of their probabilities. α which is approaching one gives the Shannon's index. When $\alpha = 0$, it is the maximum possible Shannon's index.

Comparison

All indices above consider both richness and evenness. We further compare them in the view of sample size sensitivity and difficulty of calculation [5]. Table 2 lists the comparison. Simpson's index has lower sample size sensitivity than

Shannon's index as well as Renyi's index. In this thesis, packet number of packet traces in experiments ranges from 1 to 10000000, thus we prefer Simpson's index. Furthermore, calculation of Simpson's index is simpler than Shannon's index and Renyi's index.

Table 2 Comparison between diversity indices

	Simpson's index	Shannon's index	Renyi's index
Sample size sensitivity	Low	High	High
Calculation	Simple	Moderate	Moderate

2.2 Code Coverage

Overview

Code coverage [8][9] was first mentioned in "Communications of the ACM" in 1963 by Miller and Maloney. It is a measure used in systematic software testing to describe the degree to which the source code of a program has been tested.

Software testing can be categorized simply into black-box testing and white-box testing. Black-box testing is based on what a system is required to do, while white-box testing is based on how a system operates. Coverage-based testing provides a way to quantify the degree of thoroughness of white-box testing [10].

Coverage criteria

To measure how well the program is executed by a test suite, we can use one or more coverage criteria. There are three basic coverage criteria [11][12]:

A. Function level coverage – The percentage of functions which have been called in a program.

B. Branch level coverage – The percentage of branches of control structures which have been decided in a program.

C. Line level coverage – The percentage of lines which have been executed in a program.

For example, consider the following C++ function.

```
int foo (int x, int y)
{
    int z = 0;
    if ((x>0) && (y>0))
        z=x;
    return z;
}
```

Assume this function is part of some bigger program and this program is running with some test suite. If during the execution function “foo” is called at least once, the function level coverage for “foo” is satisfied. Branch level coverage can be satisfied with test cases that call foo(1, 1), foo(1, 0) and foo(0, 0). These are necessary as in the first two cases (x > 0) evaluates to true while the third false. Line level coverage can be satisfied if foo(1, 1) is called. In this case, every line in this function is executed.

Moreover, the increasing size and complexity of software system has led to increasing challenges in evaluating code coverage. There would be scalability issues with large software for common code coverage tools. Prioritized coverage approach [13] is proposed to provide capabilities for evaluating code coverage and setting priorities for testing.

Chapter 3 Problem Statement

In this chapter, we define diversity indices to alleviate comparison between packet traces. Criterion of code coverage is also defined. Finally, we give a problem statement.

3.1 Definitions

A. Diversity index

We define six diversity indices which includes fixed-length headers and variable-length header. Each index is calculated by the formula of Simpson's index. D_{s_IP} is the diversity index for source IP of a packet trace, which means the probability that randomly select two packets from a packet trace, the source IPs are different. Thus D_{d_IP} is for destination IP, D_{s_port} for source port, D_{d_port} for destination port. Moreover, we define a mixed diversity index $D_{mix}= D_{s\_IP} * D_{d\_IP} * D_{s\_port} * D_{d\_port}$ which means the probability that randomly select two packets from a packet trace, the source IPs, destination IPs, source ports and destination ports are different simultaneously. We also define D_{app} for application header. In this thesis, we look at "http.host" in HTTP header since HTTP is popular in application layer and "http.host" means the URL. IP addresses imply hosts in a packet trace, and port numbers the applications. Source IP, destination IP, source port, destination port are the fixed-length headers, while application headers the variable-length headers.

Code coverage

We define *C* the percentage of branches of control structures which have been decided in a program,

$$\mathrm{C} = \frac{number\ of\ branches\ that\ have\ been\ decided\ in\ target\ program}{total\ number\ of\ branches\ in\ target\ program}.$$

3.2 Problem Statement

In this thesis, we anticipate that using more diverse traffic to test network devices could have larger code coverage. Thus we use packet traces with different number of packets and size of network segments to test target programs. We want to find the relationship between the defined diversity indices of packet traces and code coverage of target programs that packet traces can achieved.

The problem could be formulated as follows.

Given:

Packet traces with different number of packets and different size of network segments.

Assumption:

Packet trace with higher traffic diversity D_{s_IP}, D_{d_IP}, D_{s_port}, D_{d_port} and D_{mix} can reach larger code coverage C.

Goal:

Find the relationship between the D_{s_IP}, D_{d_IP}, D_{s_port}, D_{d_port} and D_{mix} and code coverage C.

3.3 Example

We give an example in this section. Two packet traces, 100000.pcap which has 100000 packets and 10000000.pcap which has 10000000 packets, are used here. Target program is Snort. Table 3 lists diversity indices and the code coverage these two packet traces can achieve on Snort. In this example, all diversity indices in 10000000.pcap are larger than those in 100000.pcap. The code coverage 10000000.pcap can achieve is also larger than 100000.pcap can achieve. Thus, code coverage has positive correlation to diversity indices.

Table 3 Example of problem statement

	D_{s_IP}	D_{d_IP}	D_{s_port}	D_{d_port}	D_{mix}	C
100000.pcap	0.96	0.85	0.97	0.86	0.71	24.4%
10000000.pcap	0.99	0.98	0.98	0.99	0.94	32.2%

Chapter 4 System Architecture and Proposed Methodology

In this chapter, we illustrate the system architecture for diversity index calculator and code coverage analyzer. Furthermore, we detail the methodology for calculating diversity indices and analyzing code coverage.

4.1 System Architecture

The system consists of two main components – diversity index calculator and code coverage analyzer, as shown in Fig. 1.

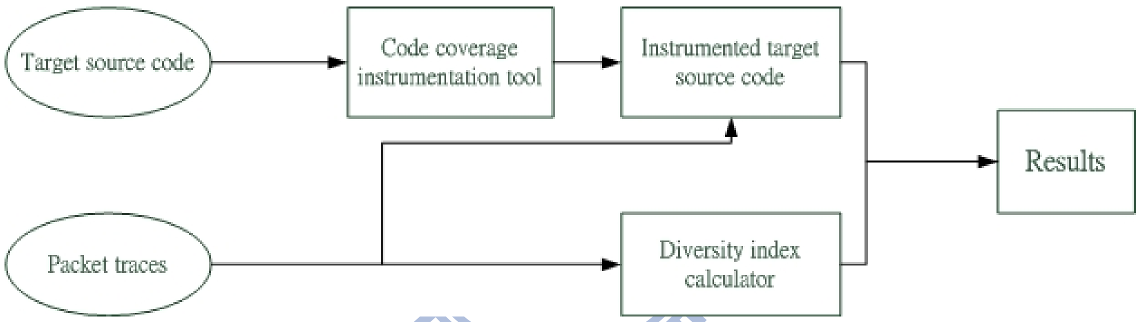
Target source code
Code coverage instrumentation tool
Instrumented target source code
Results
Packet traces
Diversity index calculator

Fig. 1 Diversity index calculator and code coverage analyzer

The purpose of diversity index calculator is to calculate the diversity of header fields selected from a packet trace. The selected header fields are extracted from the packet trace first. Then the diversity index of each header field is calculated using the Simpson's index formula. In the code coverage analyzer, the source code of the target program is instrumented first by an instrumenting tool. Packet traces are then replayed in turns as input data on the test platform. The code coverage of the target program is obtained on test completion and the results is analyzed.

4.2 Methodology

In Fig. 2, the procedure of calculating diversity index using the formula of Simpson's index is described. For a single packet trace, the chosen header fields are extracted first. The chosen header fields are source IP, destination IP, protocol number, source port, destination port, and application header. They are stored separately in files. Next, each file is sorted according to their contents, therefore identical items are grouped. Then, we calculate diversity index for each header field using Simpson's index formula $\widetilde{D}=1-D=1-\frac{\sum_{i=1}^{S} n_i(n_i-1)}{N(N-1)}$, where n_i represents the number of individuals in species i, N the total number of individuals of all species and S the number of species in a packet trace.

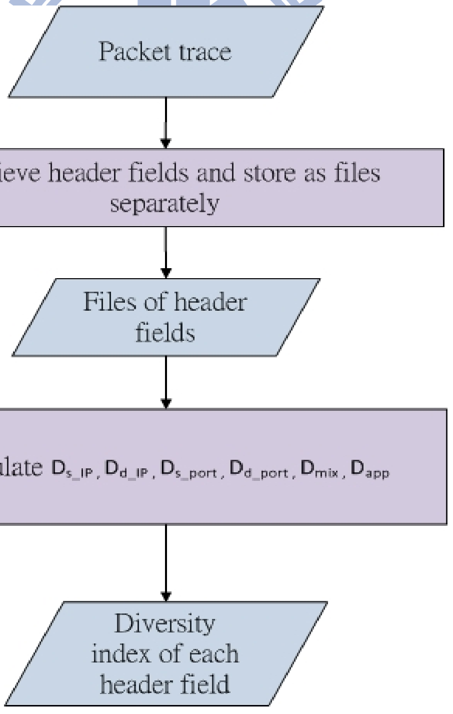
Packet trace
Retrieve header fields and store as files separately
Files of header fields
Calculate D_{s_IP}, D_{d_IP}, D_{s_port}, D_{d_port}, D_{mix}, D_{app}
Diversity index of each header field

Fig. 2 Methodology of diversity calculator

Fig. 3 shows the procedure of analyzing code coverage. The source code of the target program must be instrumented and installed on the test platform first. If the source code is of user level, Gcov is used. If kernel level, Gcov with kernel patch is applied.

Packet trace is then replayed on the test platform to test the target programs. Once the target program exits, Gcov collects and analyzes the statistics of code coverage.

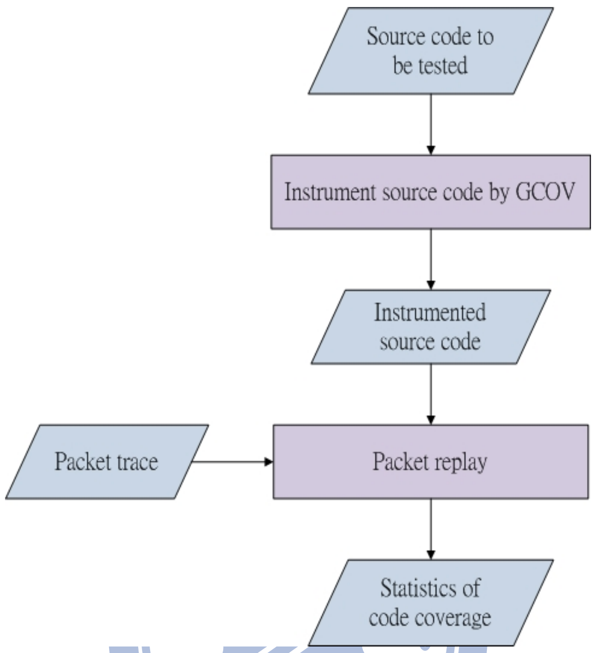

Fig. 3 Methodology of code coverage analyzer

Chapter 5 Experiments and Observations

Experiment results are shown in this chapter. Snort and Linux kernel are the target source codes. Packets traces are those with different number of packets and size of network segments.

5.1 Environment

The programs to be tested are Snort-2.9.0.5 and Linux kernel-2.6.35. Snort is user-level program while Linux kernel the kernel-level program. Both have to be instrumented by *Gcov* and installed on Ubuntu 10.10. *Gcov* is a code coverage analysis program used in concert with GCC to analyze programs. User-level program can be instrumented by *Gcov* directly, while kernel-level code need to instrumented by *Gcov* with kernel patch and module.

Packet traces used in following experiments are retrieved from two PCAP files which are named packet source 1 and packet source 2 respectively. Packet traces can be categorized by two purposes: number of packets (1.pcap, 10.pcap, 100.pcap, 1000.pcap, 10000.pcap, 100000.pcap, 1000000.pcap, 5000000.pcap and 10000000.pcap) and size of network segments (140.113.0.0/16.pcap, 140.113.243.0/24.pcap, 140.113.243.0/26.pcap, 140.113.243.0/28.cap, 140.113.249.0/24.pcap, 140.113.249.0/26.pcap, 140.113.249.0/28.cap). Two sets of packet traces are retrieved from packet source 1 and packet source 2 respectively. In the category of number of packets, packet trace with fewer packets is a subset of that with more packets. For example, 10.pcap is a subset of 100.pcap and also a subnet of 10000.pcap. In the category of size of network segments, packet trace with smaller segment is a subset of that with larger segment. For example, 140.113.243.0/28.pcap is a subset of 140.113.243.0/26.pcap and also a subnet of 140.113.0.0/16.pcap.

5.2 Experiment results

On Snort

The command used in the experiments on Snort is “snort –c <snort.conf> -r <xxx.pcap> -A full” where “snort.conf” is the configuration file of Snort and “xxx.pcap” the packet trace. Table 3 summarizes the configuration of Snort.

Table 3 Configuration of Snort-2.9.0.5

Function	Status
Decoder	Enabled
Base detection engine	Enabled
Dynamic loaded library	Enabled
Preprocessor	Enabled
Output modules	Only “log_tcpdump” enabled
Customized rule set	Snortrules-2903

Code coverage and richness increases when packet number increases, but not each of diversity indices increases.

In Fig. 4(a), code coverage *C* that packet traces of packet source 1 can achieve ranges from 19.1% to 32.2% as the number of packets increases from 1 to 10000000. Diversity indices also increase except the source IP diversity index D_{s_IP} and mixed diversity index D_{mix} when the number of packets reaches 5000000. In Fig. 4(b), richness of each header that packet traces of packet source 1 can achieve increases as the number of packets increases. The reason is that Simpson’s index considers both richness and evenness. A burst of network traffic from the same source IP leads to the decrease of evenness, which can cause the decrease of diversity indices. We also notice that when the number of packets exceeds 1000000, code coverage increases slowly. It means that the packet source 1 triggers most part of source code as it can when the packet number exceeds 1000000.

In Fig. 4(c), code coverage *C* that packet traces of packet source 2 can achieve ranges from 19.4% to 31.8% as the number of packets increases from 1 to 10000000. All diversity indices increase as the number of packets increases. In Fig. 4(d), richness of each header that packet traces of packet source 2 can achieve increases as the number of packets increases. With comparing to packet source 1, packet source 2 can achieve larger code coverage and its evenness is better than packet source 1.

Code coverage increases when network segment size becomes larger, but not each of diversity indices increases.

In Fig. 6(a), code coverage *C* that packet source 1 can achieve increases from 28.2% to 32.2% when the size of network segments becomes larger. A larger size of network segments contains more hosts, implying more richness in IP diversity.

In Fig. 6(b), code coverage *C* that packet source 2 can achieve increases from 27.5% to 31.8% when the size of network segments becomes larger. However, source IP diversity index D_{s_IP} decreases in network segment 140.113.249.0/24, which means that traffic bursts occurred in 140.113.249.0/24.pcap which resulted in worse evenness.

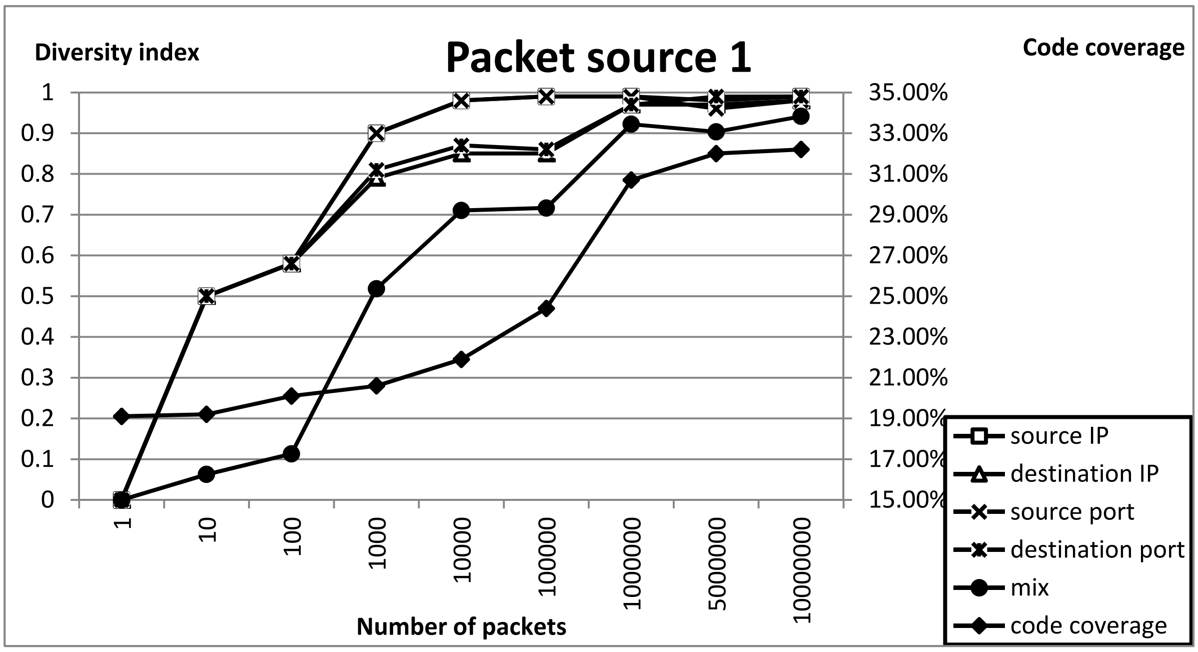

Fig. 4(a) Diversity index vs. code coverage for packet source 1 with different number of packets

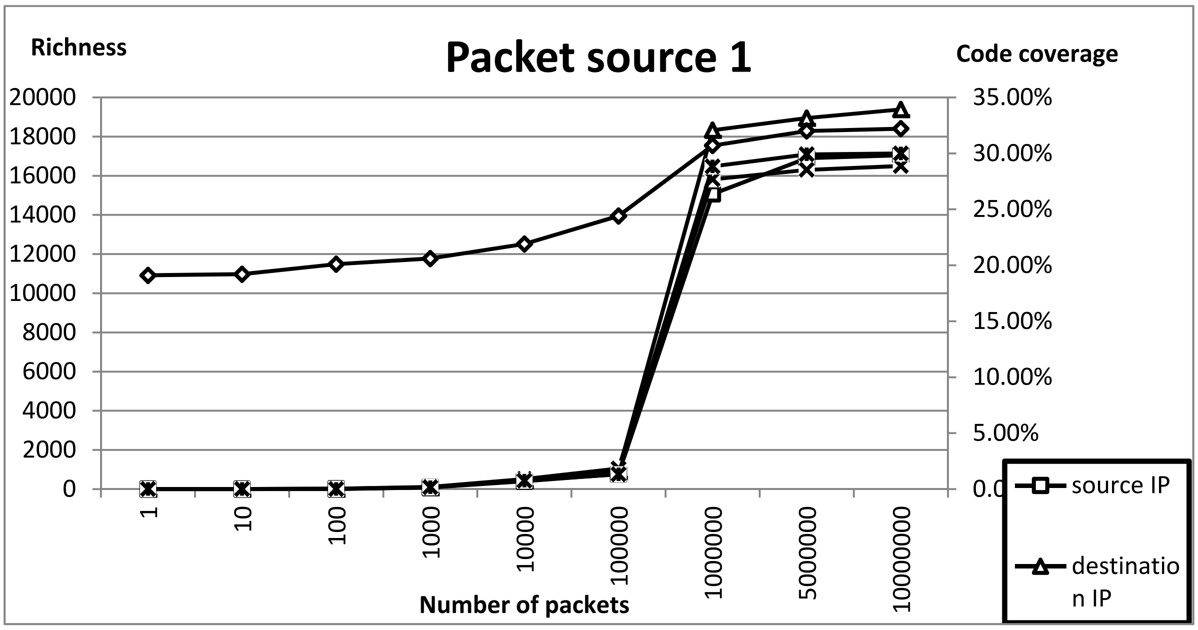

Fig. 4(b) Richness vs. code coverage for packet source 1 with different number of packets

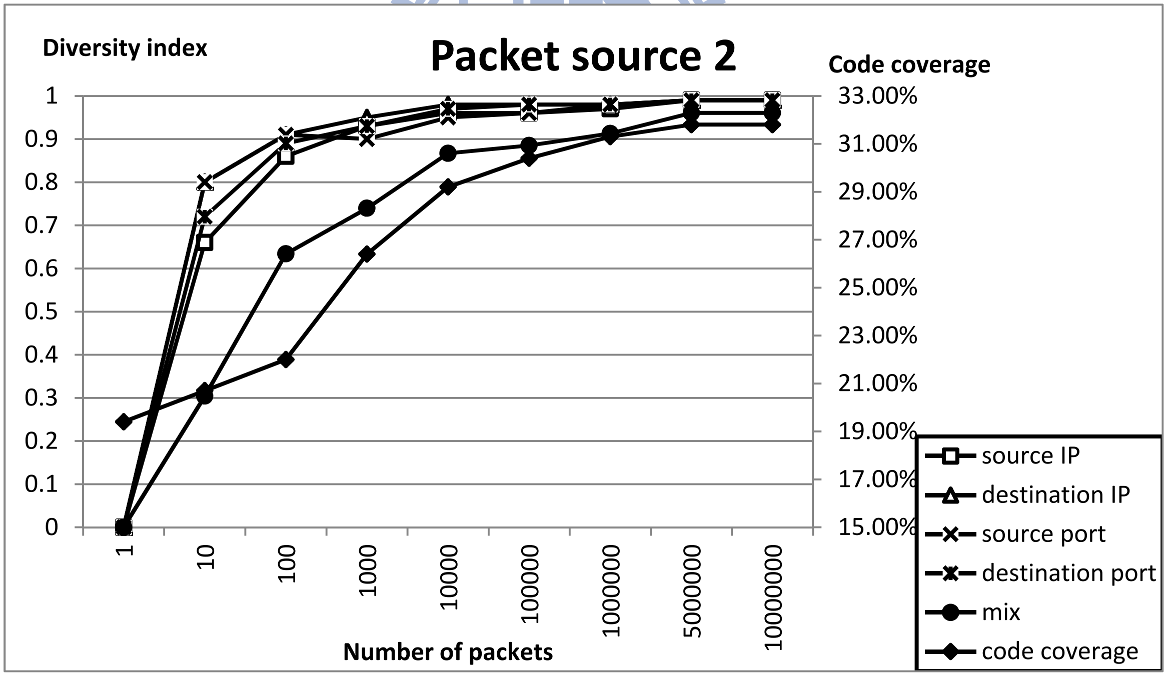

Fig. 4(c) Diversity index vs. code coverage for packet source 2 with different number of packets

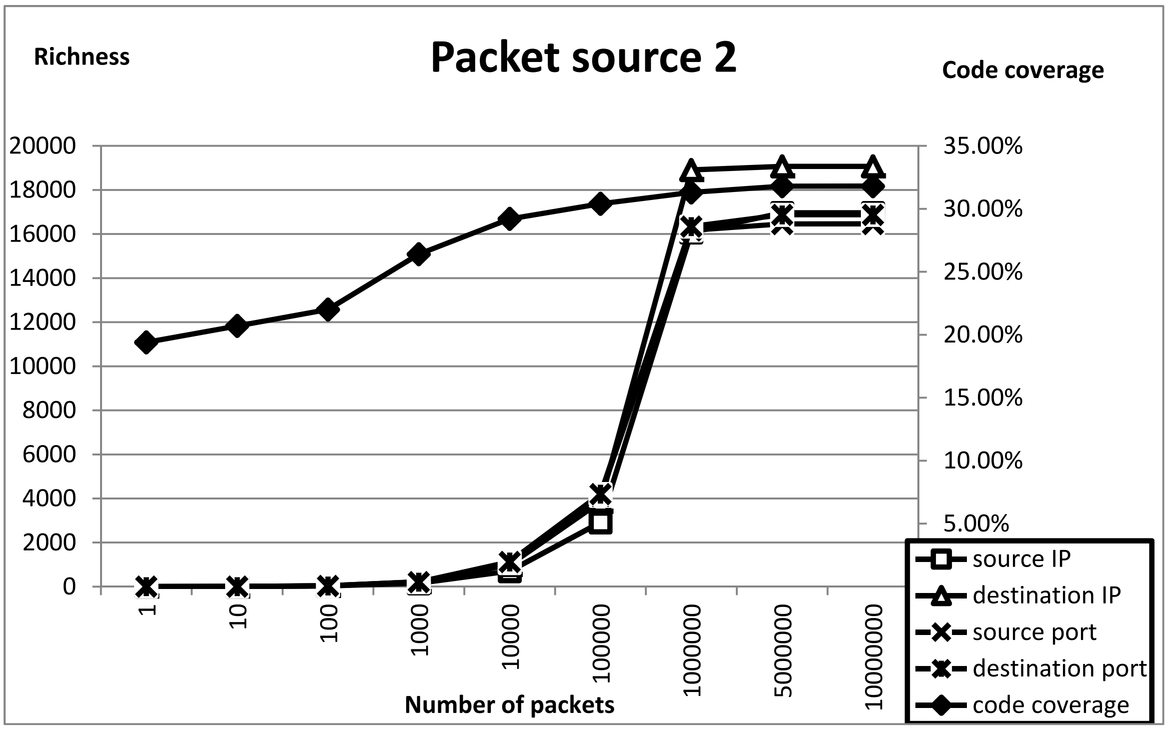

Fig. 4(d) Richness vs. code coverage for packet source 2 with different number of packets

The mixed diversity index D_{mix} has highest correlation to code coverage

We calculate the correlation coefficient between code coverage and diversity indices on Snort to find which index has highest correlation to code coverage. In Fig. 5, we can observe that the mixed diversity index D_{mix} which means the probability that randomly select two packets from a packet trace, the source IP, destination IP, source port and destination port are all different has highest correlation coefficient to code coverage comparing to other diversity indices both in packet source 1 and 2.

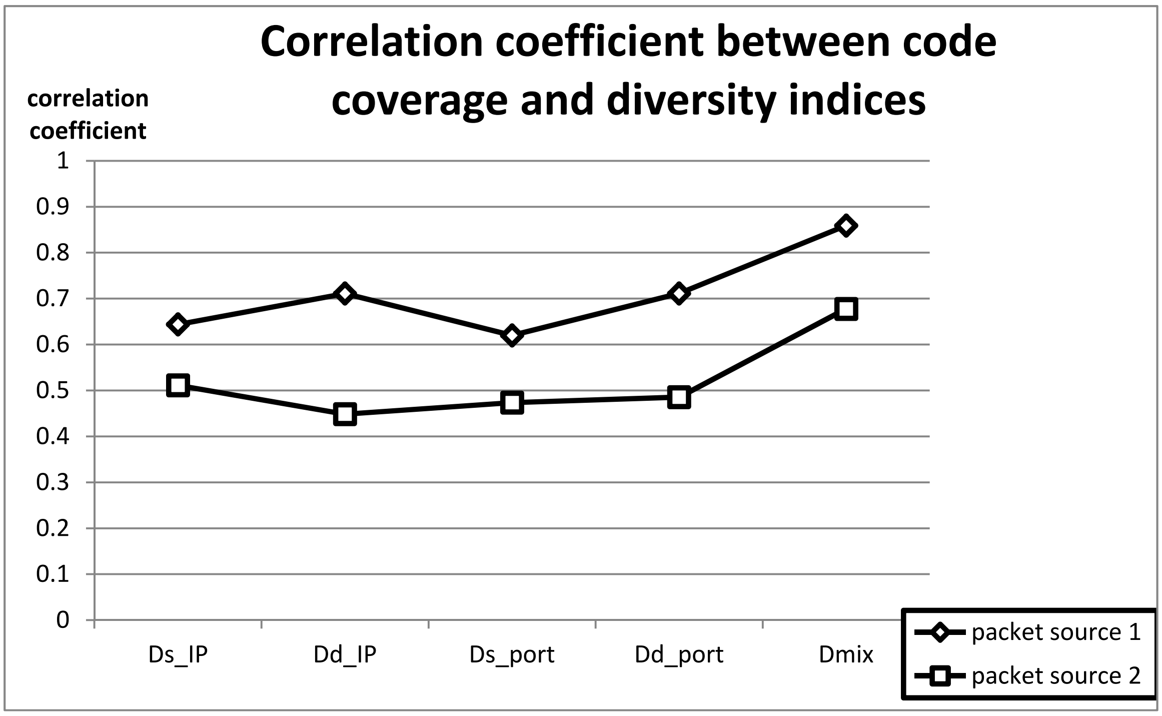

Fig. 5 Correlation coefficient between code coverage and diversity indices on Snort

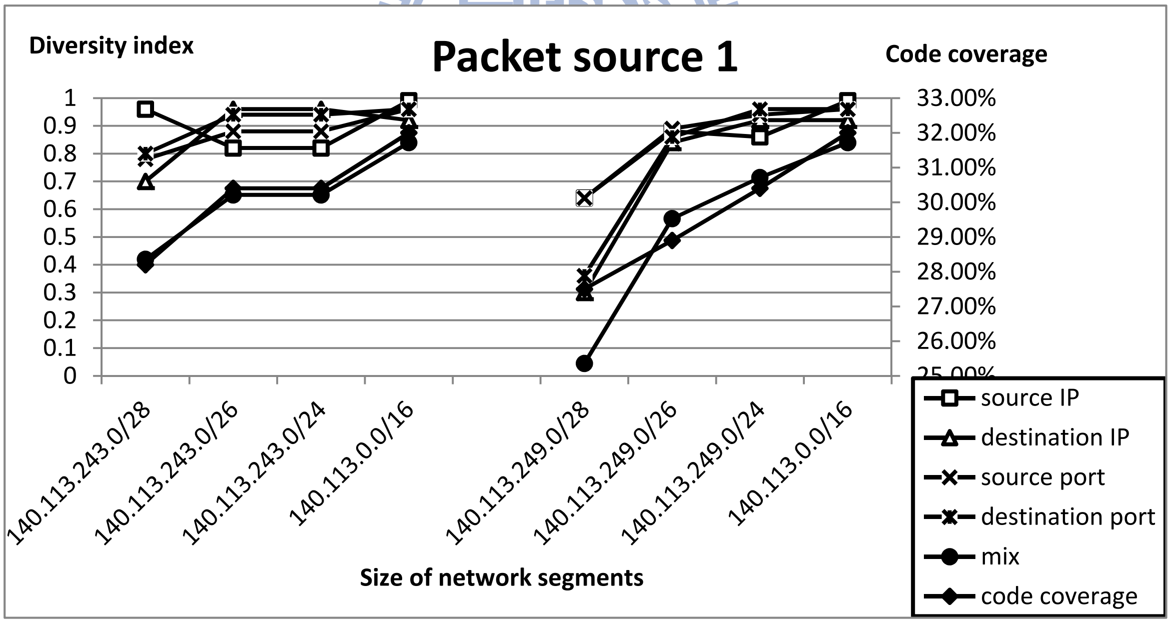

Fig. 6(a) Diversity index vs. code coverage for packet source 1 with different size of network segments

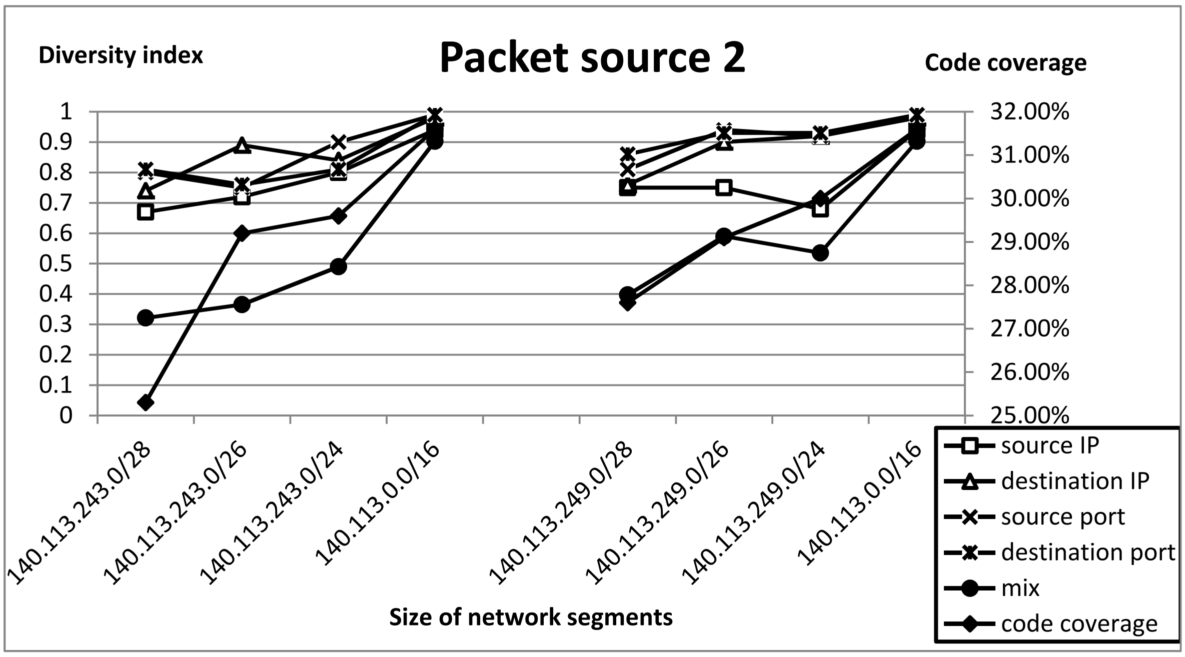

Fig. 6(b) Diversity index vs. code coverage for packet source 2 with different size of network segments in Snort

Significant increase of code coverage in specific part of source code.

In Snort, source code in the directory "HttpInspect" is used to decode user applications. Given a data buffer, HttpInspect will decode the buffer, find HTTP fields, and normalize the fields. Thus, the part of Snort source code should be greatly influenced by different packet traces. The percentage of branches of the directory "HttpInspect" to all source code is only 7%.

From Fig. 7(a), we can observe that code coverage of "HttpInspect" that packet source 1 can achieve ranges from 2.1% to 52.6%, while the code coverage of all source code from 19.2% to 32.2%.

From Fig. 7(b), we can observe that code coverage of "HttpInspect" that packet source 2 can achieve ranges from 2.1% to 53.5%, while the code coverage of all source code from 19.4% to 31.8%. Thus, packet source 2 can achieve larger code coverage when packet number reaches 10000000. We can obtain more precise analysis on the influence of different packet traces to specific part of source code.

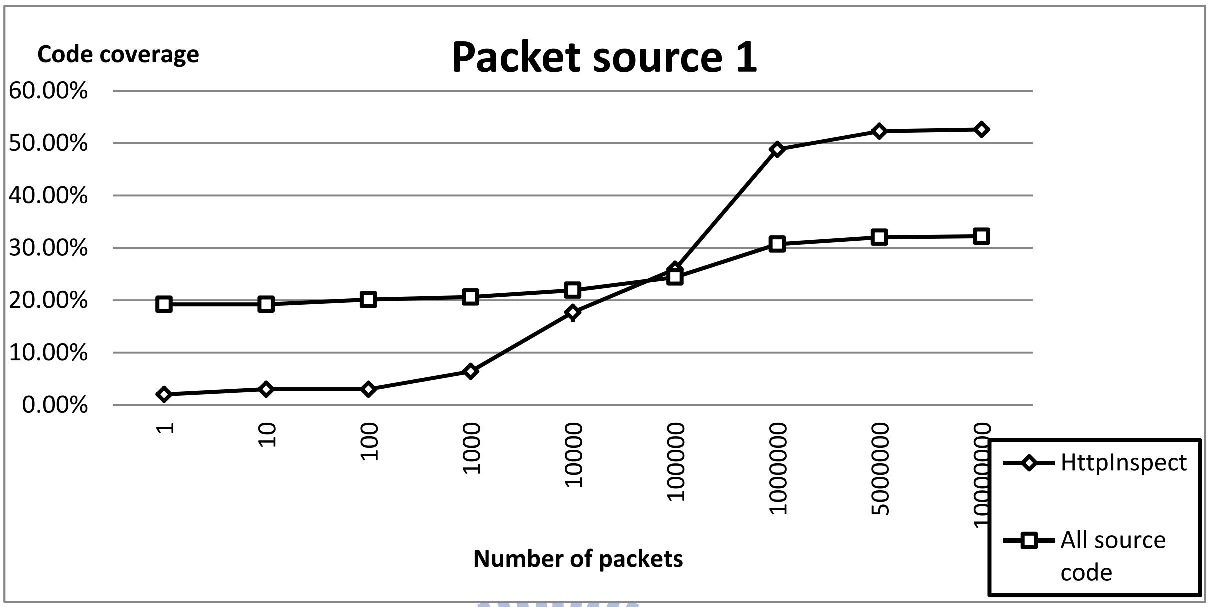

Fig. 7(a) Range of code size in Snort for packet source 1

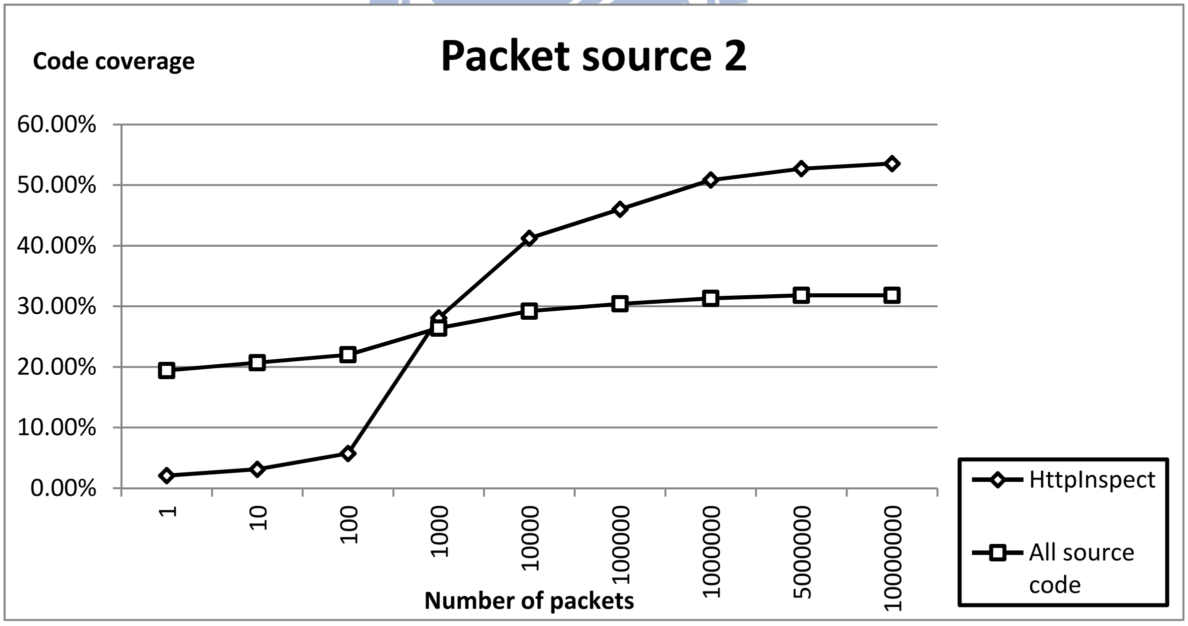

Fig. 7(b) Range of code size in Snort for packet source 2

Code coverage increases when network segment size becomes larger in the view of variable header fielder.

We can observe variable length header fields as well. The field “http.host” in HTTP header is selected. Packet traces used here are pure HTTP traffic, which are retrieved from those packet traces used in previous experiments.

In Fig. 8(a), code coverage that packet source 1 can achieve increases from 24.1% to 26.9% when the size of network segments becomes larger. However, evenness makes the diversity index decreasing at 140.113.249.0/26. However, it is clear that richness is increasing when the size of network segments becomes larger.

In Fig. 8(b), code coverage that packet source 2 can achieve increases from 22.7% to 26.8% when the size of network segments becomes larger. However, evenness makes the diversity index decreasing from 140.113.249.0/28 to 140.113.0.0/16.

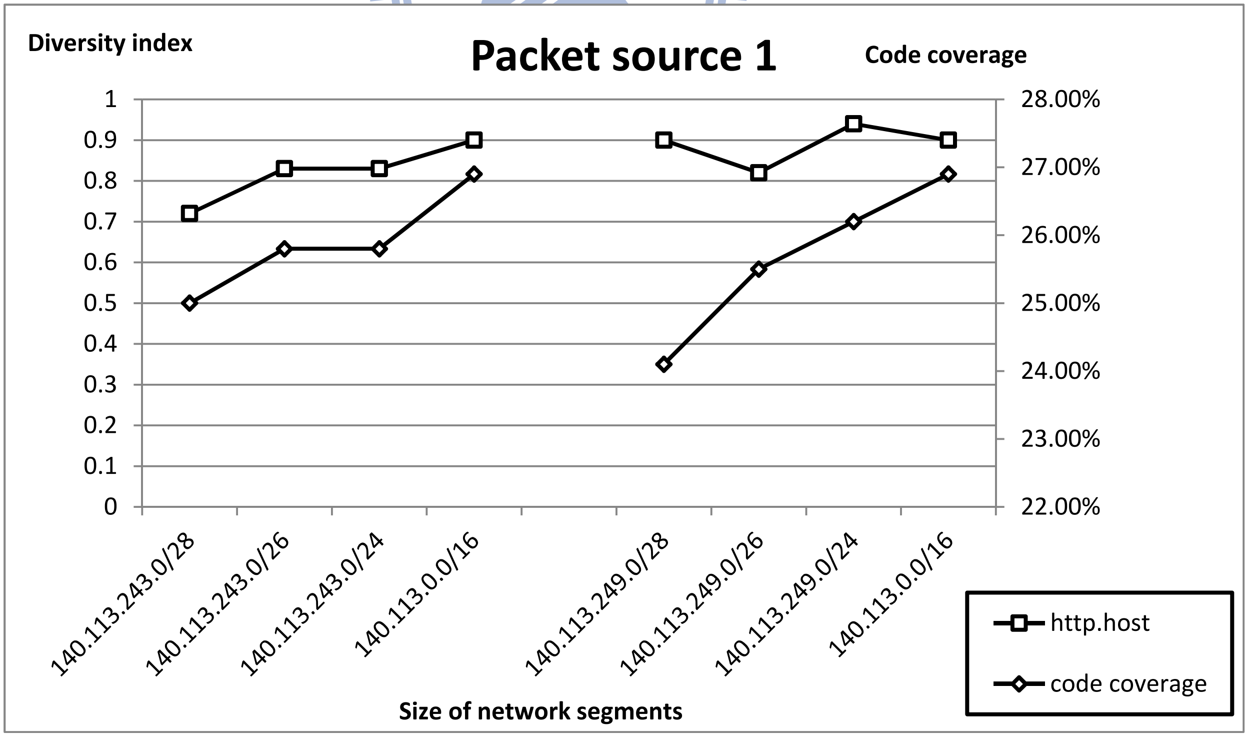

Fig. 8(a) Diversity index vs. code coverage for packet source 1 with different size of network segments

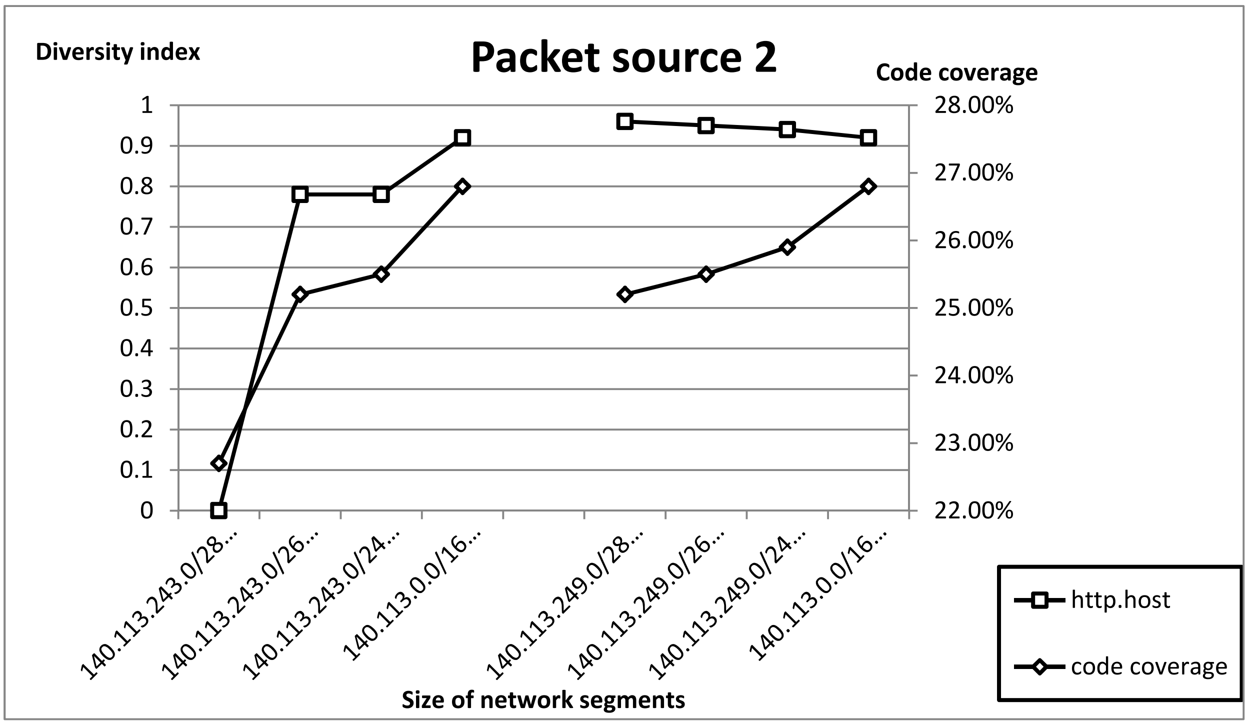

Fig. 8(b) Diversity index vs. code coverage for packet source 2 with different size of network segments

On Linux kernel

Linux kernel-2.6.35 was instrumented by *Gcov* with kernel patch and recompiled. We only observed source code under directory "/net" since the whole Linux kernel is too large and the directory /net is directly related to network traffic.

Code coverage increases when packet number increases, but not each of diversity indices increases.

Code coverage of the experiments on Linux kernel is not as that large as in Snort. In Fig. 9(a), code coverage that packet source 1 can achieve increases from 6.07% to 8.16% when the number of packets ranges from 1 to 10000000, and so do the diversity indices. It means that evenness is achieved since diversity indices are increasing (as least not decreasing) when the number of packets increases. In Fig. 9(b), code coverage that packet source 2 can achieve increases from 5.81% to 9.47% when the number of packets ranges from 1 to 10000000, and so do the diversity indices. Packet source 2 can achieve larger code coverage than packet source 1.

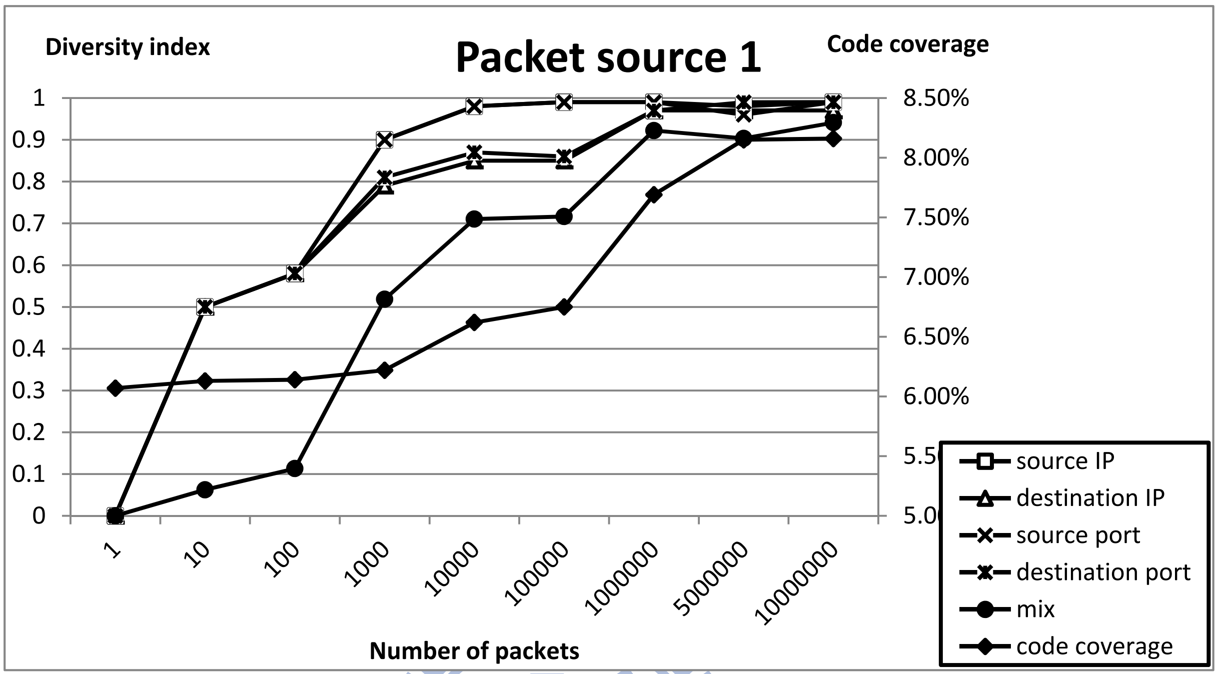

Fig. 9(a) Diversity index vs. code coverage for packet source 1 with different number of packets in Linux kernel

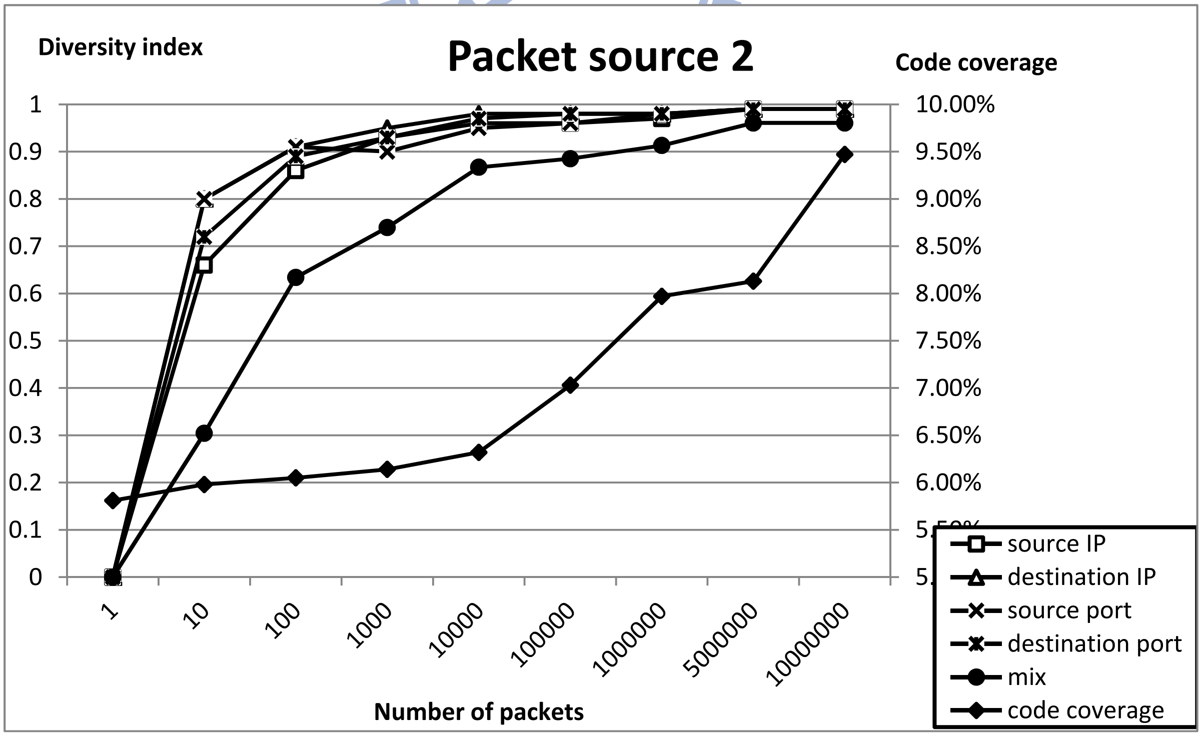

Fig. 9(b) Diversity index vs. code coverage for packet source 2 with different number of packets in Linux kernel

The mixed diversity index D_{mix} has highest correlation to code coverage

We calculate the correlation coefficient between code coverage and diversity indices on Linux kernel to find which index has highest correlation to code coverage. In Fig. 10, we can observe that the mixed diversity index D_{mix} which means the probability that randomly select two packets from a packet trace, the source IP, destination IP, source port and destination port are all different has highest correlation coefficient to code coverage comparing to other diversity indices both in packet source 1 and 2.

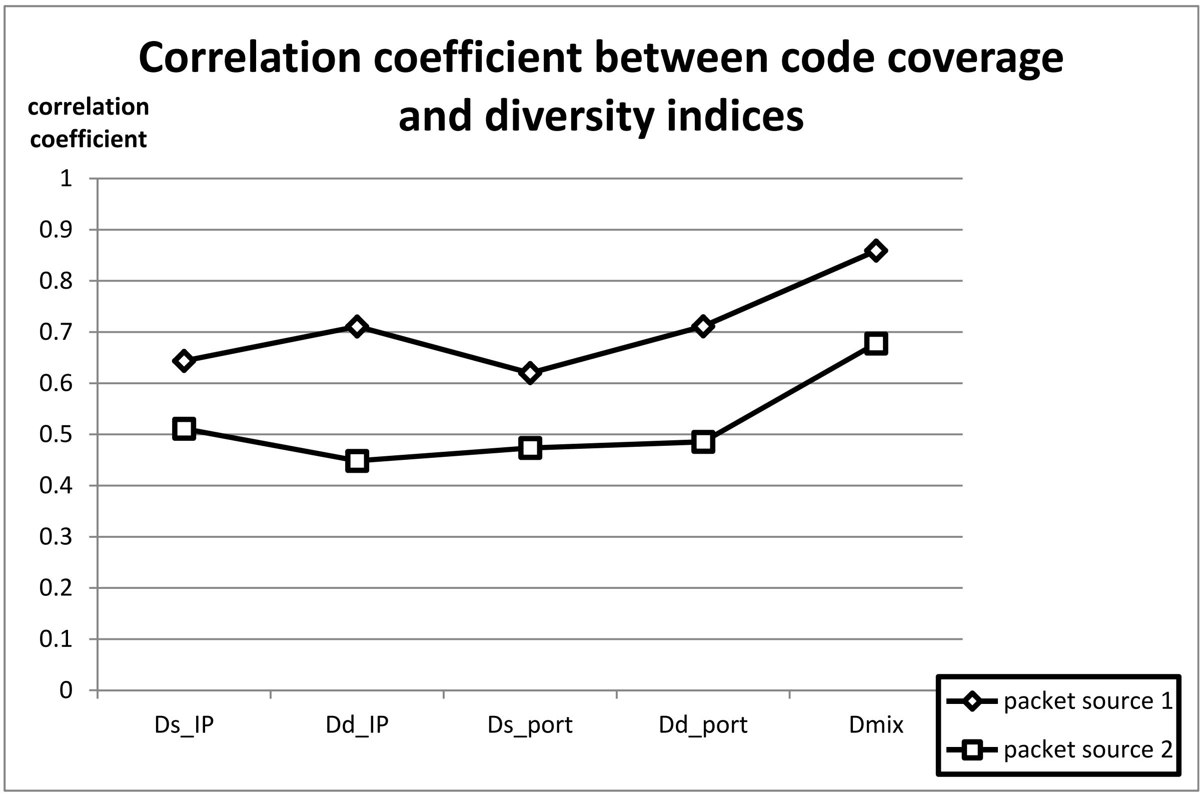

Fig. 10 Correlation coefficient between code coverage and diversity indices on Linux kernel

Code coverage increases when network segment size becomes larger, but not each of diversity indices increases.

In Fig. 11(a) and 11(b), code coverage increases when the size of network segment becomes larger. A larger size of network segments means more hosts and thus more richness in IP diversity. However, the source IP diversity index D_{S_IP} decreases at 140.113.249.0/24 in packet source 1, which means worse evenness.

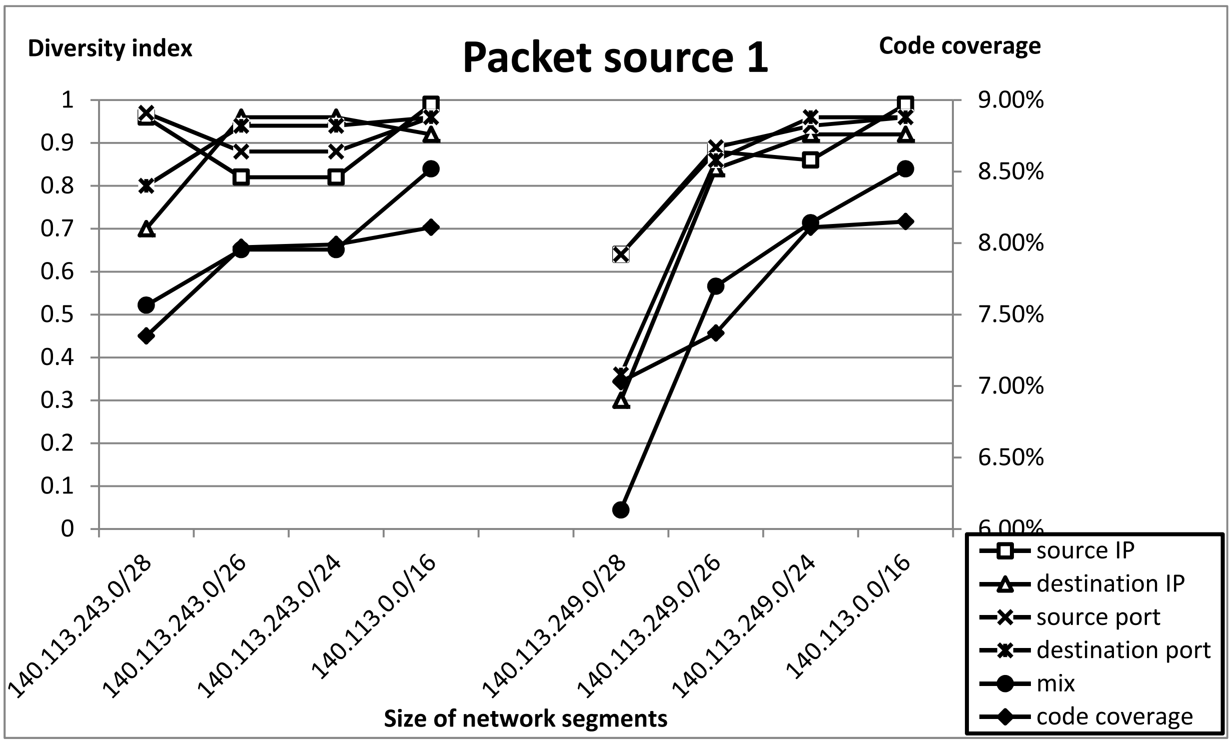

Figure 11(a) Diversity index vs. code coverage for packet source 1 with different size of network segments in Linux kernel

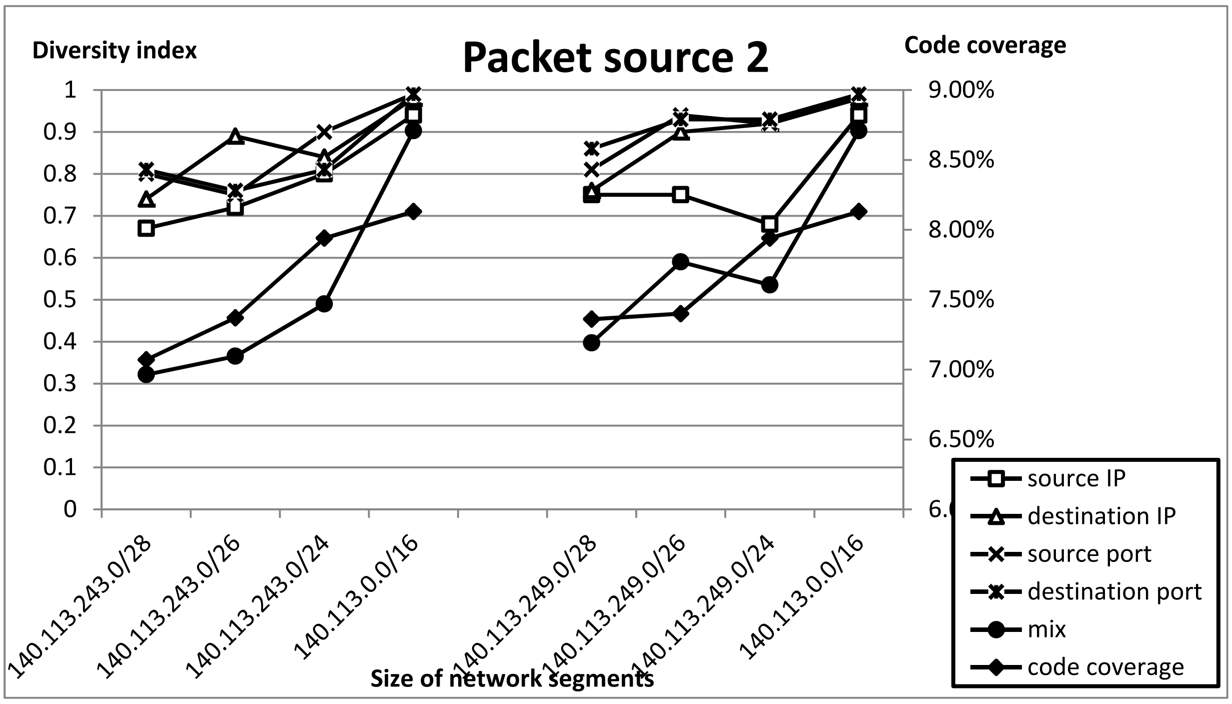

Figure 11(b) Diversity index vs. code coverage for packet source 2 with different size of network segments in Linux kernel

Chapter 6 Conclusion

In this thesis, we define diversity indices for both fixed length header fields and variable header fields to alleviate comparison between packet traces and propose a methodology for calculating diversity index and analyzing code coverage. The packet traces are with different number of packets and different size of network segments. Traffic diversity is calculated by the formula of Simpson's index which considers richness and evenness at the same time. We analyze both user level source code - Snort and kernel level source code – Linux kernel. Source code is instrumented and analyzed by *Gcov* to get its code coverage.

From experiment results, we can observe that code coverage increases when the number of packets or size of network segments of packet traces increase. For Snort, code coverage that packet source 1 can achieve is 32.2% while packet source 2

31.8%. For Linux kernel, code coverage that packet source 1 can achieve is 8.16% while packet source 2 9.47%. We can conclude that packet source 1 can achieve larger code coverage that packet source 2 in Snort, while packet source 2 can achieve larger code coverage that packet source 1 in Linux kernel. Richness of packet traces is increasing when the number of packets or size of network segments increases. However, evenness may be influenced by traffic bursts, which leads to the decreasing of some diversity indices.

In the experiments on Snort, we can see from Fig. 4 and 6 that when the number of packets exceeds 1000000, code coverage increases slowly. It means that majority of Snort code which packet traces can trigger are triggered when packet number reaches 1000000.

From Fig. 7, we can observe the significant influence of different packet traces on the specific part of source code. From Fig. 5 and 10, we can observe that the mixed the mixed diversity index D_{mix} has highest correlation coefficient to code coverage comparing to other diversity indices

In the future, we will build a more robust and real-time platform to calculate diversity indices and analyze code coverage on target source code.

References

[1] Ying-Dar Lin, I-Wei Chen, Po-Ching Lin, Chang-Sheng Chen, “On campus beta site: architecture designs, operational experience, and top product defects,” *IEEE Communication Magazine*, vol. 48, pp. 83-91, Dec. 2010.

[2] Raymond F. Dasmann, “A Different Kind of Country,” *MacMillan Company*, New York.

[3] Edward H. Simpson, ” Measurement of diversity,” Nature 163:688, 1949.

[4] Claude E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal* **27**: 379–423 and 623–656, 1948.

[5] Edward F. Menhinick, “A Comparison of Some Species-Individuals Diversity Indices Applied to Samples of Field Insects,” *Ecology*, vol. 45, pp. 859-861, 1964.

[6] Lucien M. Le Cam, "On Some Asymptotic Properties of Maximum Likelihood Estimates and Related Bayes' Estimates," University of California Publications in Statistics, 277-330, 1953.

[7] Alfred Renyi, “On the measures of entropy and information,” *Proceedings of Fourth Berkeley Symposium on Mathematics, Statistics and Probability*, vol. 1, pp. 547–561, 1961.

[8] Glenford J. Myers, “The Art of Software Testing, 2nd edition,” Wiley, 2004.

[9] Joan C. Miller, Clifford J. Maloney, "Systematic mistake analysis of digital computer programs," *Communications of the ACM*, 1963.

[10] Qian Yang, J. Jenny Li, David Weiss, “A Survey of Coverage Based Testing Tools,” *ACM Proceedings of the international workshop on Automation of software test*, 2006, pp. 99-103, 2006.

[11] Adam Kolawa, Dorota Huizinda, “Automated Defect Prevention: Best Practices in Software Testing,” *IEEE Computer Society Press*, pp.254, 2007.

[12] Martin R. Woodward, Michael A. Hennell, "On the relationship between two control-flow coverage criteria: all JJ-paths and MCDC," *Information and Software Technology*, pp. 433-440, 2006.

[13] Mechelle Gittens, Keri Romanufa, David Godwin, Jason Racicot, “All code coverage is not created equal: a case study in prioritized code coverage,”*ACM Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, 2006.